Data Request Timing

Comparison of 3 protocols

Mar 5, 1991

The VME Local Station software supports three data acquisition protocols. The first is the Classic protocol, which was the original protocol developed in 1982. On the token ring network, it uses a special SAP to distinguish it from Acnet header-based protocols. The other two protocols supported are based upon the use of the Acnet header that supports general task-task communication across a single network. This note describes the performance by the local station software in its support of all three protocols.

The Classic protocol is based upon listypes and idents, two abstract specifiers which characterize control system data requests. The design is targeted for distributed systems, in which a single network of local stations connect to the control system signals of a part of an accelerator and contain a local database for their own parts. Because of this, these systems can operate in the absence of a centralized host system. Any host can participate by using the data request protocols.

The second is the DZero protocol, which was developed to satisfy the needs of the D0 control system. It is an evolution of the Classic protocol in that its design is also based upon listypes and idents.

The other is the Accelerator protocol, developed for use with the Fermilab accelerator control system. It was designed to work only with a centralized database. The user program running on a Vax console is given a suite of `DPxxx` routines that hide the central database accesses used to build a data request.

Two sets of timings were made for each of the three protocols. One was made from the requester's point of view; the other was made from the replier's point of view. The timing was measured by software using a timer of 0.5 msec resolution in the first case and by observing signals available on a front panel connector of the Crate Utility board that show each task's activity in the second.

The example used in the measurements was a data request for readings and setting words of a number of analog channels from one other local station. The number was varied to get the incremental timing on a per channel basis. The test programs are local station console page applications designed for testing each protocol.

The first test measures the time from just before a one-shot request is made until the time that the answers are available in the application's data arrays. For a minimal data request, the timing is about 5 msec independent of the protocol. The timings for a number of channels > 1 is as follows:

Protocol	*#chans*	*One-shot, msec*	*Per channel, µs*
Classic	1	5.5	
	51	10.0	90
DZero	1	5.5	
	51	8.5	60
Accel	1	4.5	
	51	23.5	380

A reason for the longer time per channel in the Classic protocol compared to the DZero protocol is that the ReqData routine called by the test application includes a "data server" functionality for local requests, so there is work to be done before the data request message is prepared for the network. In the other two cases, the test program prepares the network message before the timing starts.

The second test measures the additional time spent in the Update Task due to updating the answers in fulfilling the repetitive data request. It does not include any time for transmitting the results across the network. It represents only the additional load on the cpu in producing a set of answers to the data request.

Protocol	*#chans*	*15 Hz, msec*	*Per channel, µs*
Classic	1	0.4	
	51	0.6	4
DZero	1	0.4	
	51	0.6	4
Accelerator	1	0.4	
	51	2.0	32

The increased time used by the accelerator protocol is because it was not designed for efficient processing. The key concept that is missing is that of specifying an array of idents to be processed under a given listype.